Design and Implementation of Parallel Self-Timed Adder Using VHDL

[1]Dr. Ahmed Sajjad Khan,[2] Mohammad Hassan,[3]Syed Aiman, [4]Shreya Raut, [5]Shefali Gedam
[1]Professor,[2] B.E Student,[3] B.E Student, [4] B.E Student, [5] B.E Student
[1]Department of Electronics and Telecommunication Engineering,
[1]Anjuman College of Engineering and Technology, Nagpur, India

Abstract-As technology scales down into the lower nanometer values power, delay, area and frequency becomes important parameters for the analysis and design of any circuits. This brief presents a parallel single-rail self-timed adder. It is based on a recursive formulation for performing multi-bit binary addition. The operation is parallel for those bits that do not need any carry chain propagation. Thus, the design attains logarithmic performance over random operand conditions without any special speedup circuitry or look-ahead schema. A practical implementation is provided along with a completion detection unit. The implementation is regular and does not have any practical limitations of high fan-outs. A high fan-in gate is required though but this is unavoidable for asynchronous logic and is managed by connecting the transistors in parallel. Simulations have been performed using industry standard toolkit that verify the practicality and superiority of the proposed approach over existing asynchronous adders.

IndexTerms—CMOS design, digital arithmetic Binary adders, Recursive adder.

I. INTRODUCTION (HEADING 1)

The most important operation of processor that can perform is binary addition. The operation is based on the recursive formulation for performing multi-bit binary addition. The binary subtraction is another arithmetic operation. The serial adder has slower, uses shift registers, requires one full adder circuit, Time required for addition depends on number of bits and it is sequential circuit. But the parallel adder is Faster, It uses registers with parallel load capacity, No. of full adder circuit is equal to No. of bits in binary adder and time required does not depend on the number of bits. Here half adders are used instead of full adder .Here the operation is parallel for those bits so that no need any carry chain propagation. In cellular phones, PDA and other high performance, the clock-less chip processor approach must be needed. Clock-less processors also called asynchronous or self -timed. In self timed system, that doesn't use the oscillating crystal that serves as the regularly "ticking" clock [2]. So the clock less chip will run faster than clock chip in order to avoid the need of clock tick. Also in clock-less chip, the major advantage that have of low electromagnetic interference. Asynchronous circuits do not assume any quantization of time. In absence of clocks, the logic flow in asynchronous circuit is controlled by request –acknowledgement handshaking protocol. But handshaking block for small elements, such as bit adders and bit subtractions are expensive. So these managed using dual rail carry propagation in adders and borrow propagation in subtractor.

Binary addition is the single most important operation that a processor performs. Most of the adders have been designed for synchronous circuits even though there is a strong interest in clock less circuits [1].Asynchronous circuits do not assume any quantization of time. Therefore, they hold great potential for logic design as they are free from several problems of clocked (synchronous) circuits. In principle, logic flow in asynchronous circuits is controlled by wave pipelining (or maximal rate pipelining) is a technique that can apply pipelined inputs before the outputs are stabilized [7].

II. BACKGROUND

There are a myriad designs of binary adders and we focus here on asynchronous self-timed adders. Self-timed refers to logic circuits that depend on and/or engineer timing assumptions for the correct operation. Self-timed adders have the potential to run faster averaged for dynamic data, as early completion sensing can avoid the need for the worst case bundled delay mechanism of synchronous circuits. They can be further classified as follows.

A. Pipelined Adders Using Single-Rail Data Encoding

The asynchronous Req/Ack handshake can be used to enable the adder block as well as to establish the flow of carry signals. In most of the cases, a dual-rail carry convention is used for internal bitwise flow of carry outputs. These dual-rail signals can represent more than two logic values (invalid, 0, 1), and therefore can be used to generate bit-level acknowledgment when a bit operation is completed. Final completion is sensed when all bit Ack signals are received (high). The carry-completion sensing adder is an example of a pipelined adder [8], which uses full adder (FA) functional blocks adapted for dual-rail carry. On the other hand, a speculative completion adder is proposed in [9]. It uses so-called abort logic and early completion to select the proper completion response from a number of fixed delay lines. However, the abort logic implementation is expensive due to high fan-in requirements.

B. Delay Insensitive Adders Using Dual-Rail Encoding

Delay insensitive (DI) adders are asynchronous adders that assert bundling constraints or DI operations. Therefore, they can correctly operate in presence of bounded but unknown gate and wire delays [2]. There are many variants of DI adders, such as DI ripple carry adder (DIRCA) and DI carry look-ahead adder (DICLA). DI adders use dual-rail encoding and are assumed to increase complexity. Though dual-rail encoding doubles the wire complexity, they can still be used to produce circuits nearly as efficient as that of the single-rail variants using dynamic logic or nMOS only designs. An example 40 transistors per bit DIRCA adder is presented in [8] while the conventional CMOS RCA uses 28 transistors.

Similar to CLA, the DICLA defines carry propagate, generate, and kill equations in terms of dual-rail encoding [8]. They do not connect the carry signals in a chain but rather organize them in a hierarchical tree. Thus, they can potentially operate faster when there is long carry chain. A further optimization is provided from the observation that dual rail encoding logic can benefit from settling of either the 0 or 1 path. Dual-rail logic need not wait for both paths to be evaluated. Thus, it is possible to further speed up the carry look-ahead circuitry to send carry-generate/carry-kill signals to any level in the tree. This is elaborated in [8] and referred as DICLA with speedup circuitry (DICLASP).

III. PARALLEL SELF-TIMED ADDER

In this section, the architecture and theory behind PASTA is presented. The adder first accepts two input operands to perform half additions for each bit. Subsequently, it iterates using earlier generated carry and sums to perform half-additions repeatedly until all carry bits are consumed and settled at zero level.

The general architecture of the adder is shown in Fig. 1. The selection input for two-input multiplexers corresponds to the Req handshake signal and will be a single 0 to 1 transition denoted by SEL. It will initially select the actual operands during SEL = 0 and will switch to feedback/carry paths for subsequent iterations using SEL = 1. The feedback path from the HAs enables the multiple iterations to continue until the completion when all carry signals will assume zero values.

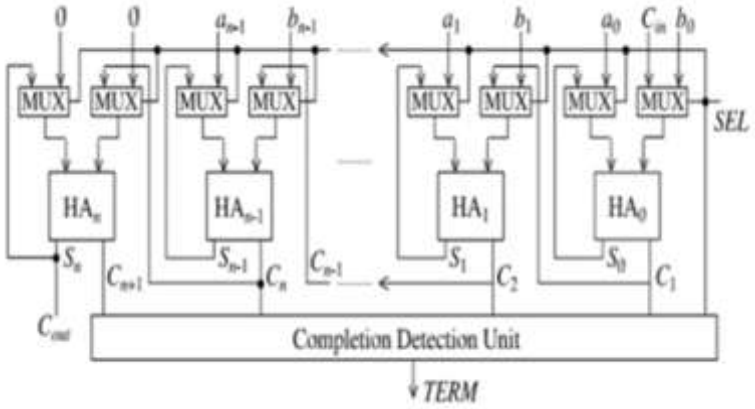

Fig.1 General Block Diagram of PASTA

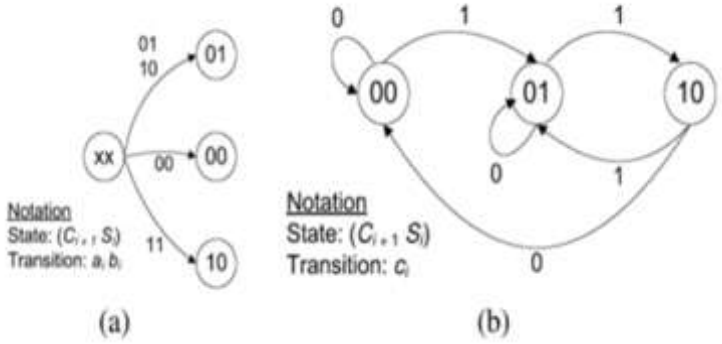

Fig.2 State diagrams for PASTA. (a) Initial phase. (b) Iterative phase.

In Fig. 2, two state diagrams are drawn for the initial phase and the iterative phase of the proposed architecture. Each state is represented by (Ci+1 Si) pair where Ci+1, Si represents carry out and sum values, respectively, from the ith bit adder block. During the initial phase, the circuit merely works as a combinational HA operating in fundamental mode. It is apparent that due to the use of HAs instead of FAs, state (11) cannot appear. During the iterative phase (SEL = 1), the feedback path through multiplexer block is activated. The carry transitions (Ci) are allowed as many times as needed to complete the recursion. From the definition of fundamental mode circuits, the present design cannot be considered as a fundamental mode circuit as the input–outputs will go through several transitions before producing the final output. It is not a Muller circuit working outside the fundamental mode either as internally; several transitions will take place, as shown in the state diagram. This is analogous to cyclic sequential circuits where gate delays are utilized to separate individual states [4].

Let Sji and Cji+1 denote the sum and carry, respectively, for ith bit at the jth iteration. The initial condition (j= 0) for addition is formulated as follows:

$$S_i^0 = a_i \oplus b_i$$

$$C_{i+1}^0 = a_i b_i. \qquad (1)$$

The jTh iteration for the recursive addition is formulated by

$$S_i^j = S_i^{j-1} \oplus C_i^{j-1}, \quad 0 \le i < n \qquad (2)$$

$$C_{i+1}^j = S_i^{j-1} C_i^{j-1}, \quad 0 \le i \le n. \qquad (3)$$

The recursion is terminated at kth iteration when the followingcondition is met:

$$C_n^k + C_{n-1}^k + \cdots + C_1^k = 0, \quad 0 \le k \le n. \qquad (4)$$

Now, the correctness of the recursive formulation is inductively proved as follows. Theorem 1: The recursive formulation of (1)–(4) will produce correct sum for any number of bits and will terminate within a finite time. Proof: We prove the correctness of the algorithm by induction on the required number of iterations for completing the addition (meeting the terminating condition).Basis: Consider the operand choices for which no carry propagation is required, i.e., C0i= 0 for i, i [0.n]. The proposed formulation will produce the correct result by a single-bit computation time and terminate instantly as (4) is met. Induction: Assume that Cki+1_= 0 for some ith bit at kth iteration. Let l be such a bit for which Ckl+1= 1. We show that it will be successfully transmitted to next higher bit in the (k + 1)th iteration. As shown in the state diagram, the kth iteration of lth bit state(Ckl+1, Skl) and (l + 1)th bit state (Ckl+2, Skl+1) could be in any of (0, 0), (0, 1), or (1, 0) states. As Ckl+1= 1, it implies that Skl= 0. Hence, from (3), Ck+1l+1= 0 for any input condition between0 to l bits. We now consider the (l + 1)th bit state (Ckl+2, Skl+1) for kth iteration. It could also be in any of (0, 0), (0, 1), or (1, 0) states. In (k+1)th iteration, the (0, 0) and (1, 0) states from the kth iteration will correctly produce output of (0, 1) following (2) and (3). For(0, 1) state, the carry successfully propagates through this bit level following (3).Thus, all the single-bit adders will successfully kill or propagate the carries until all carries are zero fulfilling the terminating condition. The mathematical form presented above is valid under the condition that the iterations progress synchronously for all bit levels and the required input and outputs for a specific iteration will also being synchrony with the progress of one iteration. In the next section, we present an implementation of the proposed architecture which is subsequently verified using simulations.

IV. Implementation Of Pasta

A CMOS implementation for the recursive circuit is shown in Fig.3. For multiplexers and AND gates we have used Xilinx ISE implementations while for the XOR gate we have used the faster ten transistor implementation based on transmission gate XOR to match the delay with AND gates [4]. The completion detection following (4) is negated to obtain an active high completion signal (TERM). This requires a large fan-in n-input NOR gate. Therefore, an alternative more practical pseudo-nMOS ratioed design is used. The resulting design is shown in Fig. 3(d). Using the pseudo-nMOS design, the completion unit avoids the high fan-in problem as all the connections are parallel. The pMOS transistor connected to VDD of these ratioed design acts as a load register, resulting in static current drain when some of the nMOS transistors are on simultaneously. In addition to the Ci s, the negative of SEL signal is also included for the TERM signal to ensure that the completion cannot be accidentally turned on during the initial selection phase of the actual inputs. It also prevents the pMOS pull up transistor from being always on. Hence, static current will only be flowing for the duration of the actual computation.

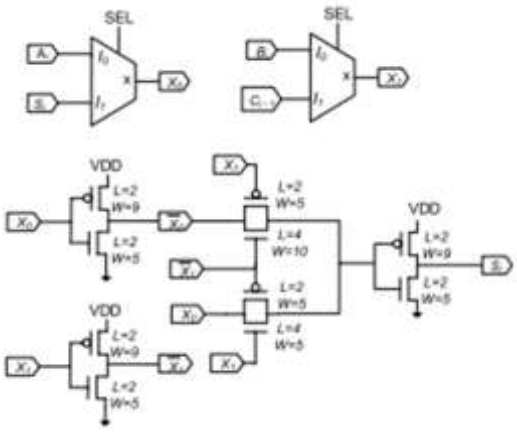

Fig.3 Single-bit sum module

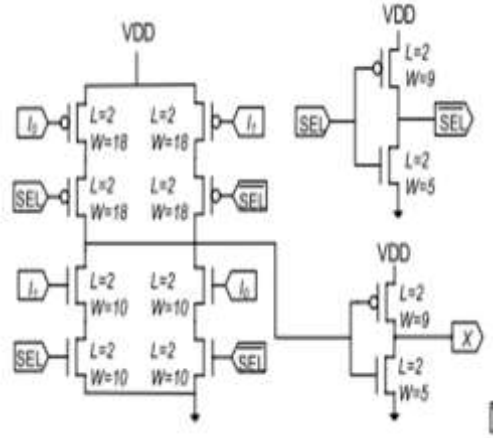

Fig.4 2 × 1 MUX for the 1 bit adder

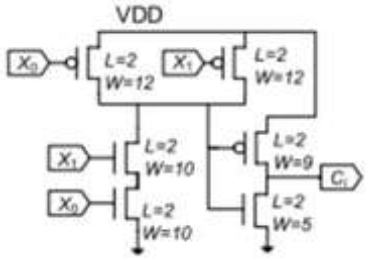

Fig.5 Single-bit carry module

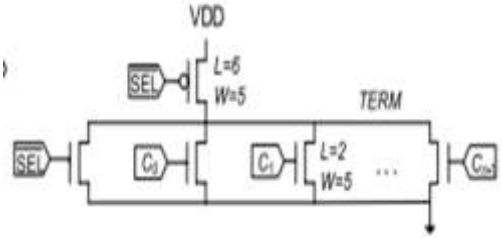

Fig.6 Completion signal detection circuit

V. Simulation Results

The corresponding simulation results of the PASTA adders are shown below. All the synthesis and simulation results are performed using Very High Speed Integrated Circuit (VHSIC) HDL. The synthesis and simulation are performed on Xilinx ISE 14.4. The simulation results are shown below figures. We have implemented PASTA for 4 bit, 8 bit, 16 bit, 32 bit, 64 bit and 128 bit addition of two binary numbers. Here, the synthesis and simulation of two 64bit binary numbers additions is depicted in Fig.7 and Fig.8 respectively.

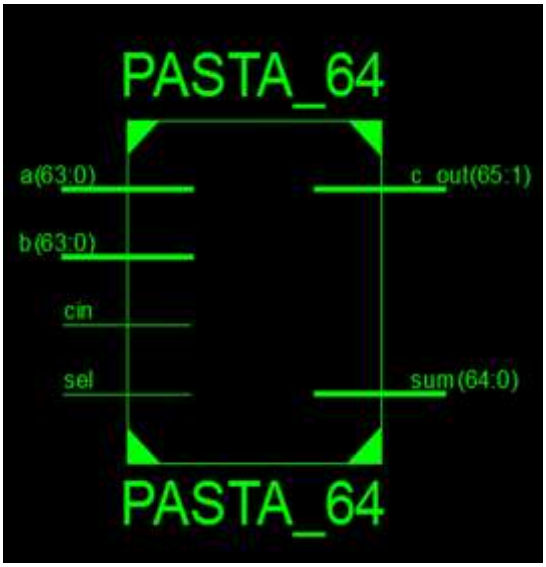

Fig.7 RTL view of 64 bit PASTA

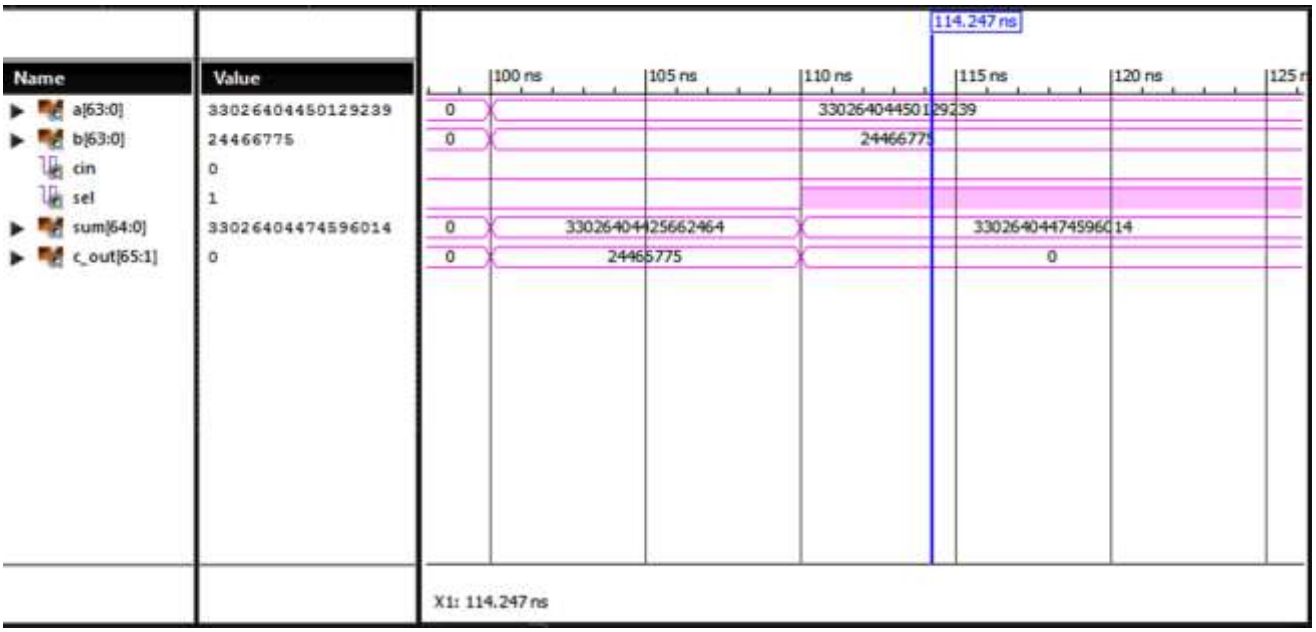

Fig.8 Simulation Result of 64 bit PASTA

VI. Conclusion

In this paper, we have presented an efficient implementation of PASTA. The architectural design and VLSI implementations using VHDL are presented. The design achieves a very simple n-bit adder that is area and interconnection-wise equivalent to the simplest adder namely the RCA. Moreover, the circuit works in a parallel manner for independent carry chains, and thus achieves logarithmic average time performance over random input values. The completion detection unit for the proposed adder is also practical and efficient. Simulation results are used to verify the advantages of the proposed approach.

References

[1] M. Z. Rahman, L. Kleeman, and M. A. Habib, "Recursive Approach to the Design of a Parallel Self-Timed Adder," IEEE Transactions On Very Large Scale Integration (VLSI) Systems, 1063-8210 © 2014 IEEE.

[2] K. R Naru and K. Kotaiah , "Design of a Parallel Self-Timed Adder Using Recursive Approach", International Journal & Magazine of Engineering, Technology, Management and Research, Vol. No: 2 (2015), Issue No: 7 (July).

[3] D. Geer, "Is it time for clock-less chips? [Asynchronous processor chips]," IEEE Comput., vol. 38, no. 3, pp. 18–19, Mar. 2005.

[4] J. Sparsø and S. Furber, Principles of Asynchronous Circuit Design.Boston, MA, USA: Kluwer Academic, 2001.

[5] P. Choudhury, S. Sahoo, and M. Chakraborty, "Implementation of basicarithmetic operations using cellular automaton," in Proc. ICIT, 2008,pp. 79–80.

[6] M. Z. Rahman and L. Kleeman, "A delay matched approach forthe design of asynchronous sequential circuits," Dept. Comput.Syst.Technol., Univ. Malaya, Kuala Lumpur, Malaysia, Tech. Rep. 05042013,2013.

[7] M. D. Riedel, "Cyclic combinational circuits," Ph.D. dissertation, Dept.Comput. Sci., California Inst. Technol., Pasadena, CA, USA, May 2004.

[8] R. F. Tinder, Asynchronous Sequential Machine Design and Analysis:A Comprehensive Development of the Design and Analysis of Clock-Independent State Machines and Systems. San Mateo, CA, USA:Morgan, 2009.

[9] W. Liu, C. T. Gray, D. Fan, and W. J. Farlow, "A 250- MHz wavepipelined adder in 2-µm CMOS," IEEE J. Solid-State Circuits, vol. 29,no. 9, pp. 1117–1128, Sep. 1994.
[10] F.-C. Cheng, S. H. Unger, and M. Theobald, "Self timed carry-look ahead adders," IEEE Trans. Comput., vol. 49, no. 7, pp. 659–672, Jul. 2000.
[11] S. Nowick, "Design of a low-latency asynchronous adder using speculative completion," IEE Proc. Comput. Digital Tech., vol. 143, no. 5,pp. 301–307, Sep. 1996.
[12] N. Weste and D. Harris, CMOS VLSI Design: A Circuits and SystemsPerspective. Reading, MA, USA: Addison-Wesley, 2005.
[13] C. Cornelius, S. Koppe, and D. Timmermann, "Dynamic circuit techniquesin deep submicron technologies: Domino logic reconsidered," inProc. IEEE ICICDT, Feb. 2006, pp. 1–4.
[14] M. Anis, S. Member, M. Allam, and M. Elmasry, "Impact of technologyscaling on CMOS logic styles," IEEE Trans. Circuits Syst., AnalogDigital Signal Process., vol. 49, no. 8, pp. 577–588, Aug. 2002.